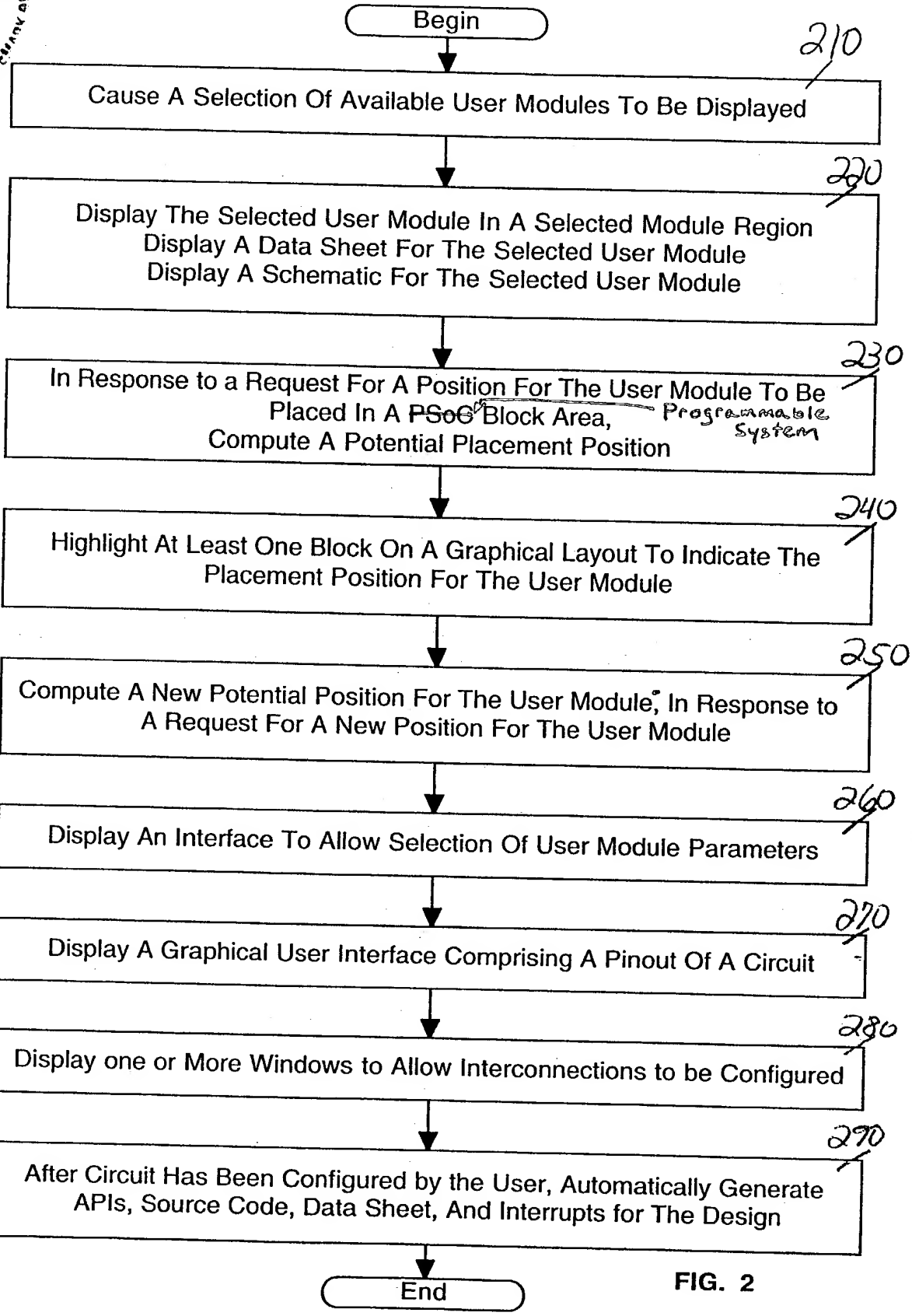

FIG. 2

(Continued)

```
        db      27h, 00h        ;ADCINC12_1_CounterCR0
        db      25h, 00h        ;ADCINC12_1_CounterDR1
        db      26h, 00h        ;ADCINC12_1_CounterDR2
;   Instance name ADCINC12_1, Block Name TMR(DBA00)
        db      23h, 00h        ;ADCINC12_1_TimerCR0
        db      21h, 00h        ;ADCINC12_1_TimerDR1
        db      22h, 00h        ;ADCINC12_1_TimerDR2
; Instance name Counter16_1, User Module Counter16
;   Instance name Counter16_1, Block Name CNTR16_LSB(DBA02)
        db      2bh, 00h        ;Counter16_1_CONTROL_LSB_REG
        db      29h, 80h        ;Counter16_1_PERIOD_LSB_REG
        db      2ah, 64h        ;Counter16_1_COMPARE_LSB_REG
;   Instance name Counter16_1, Block Name CNTR16_MSB(DBA03)
        db      2fh, 00h        ;Counter16_1_CONTROL_MSB_REG
        db      2dh, 00h        ;Counter16_1_PERIOD_MSB_REG
        db      2eh, 00h        ;Counter16_1_COMPARE_MSB_REG
; Instance name DAC8_1, User Module DAC8
;   Instance name DAC8_1, Block Name LSB(ASB11)
;   Instance name DAC8_1, Block Name MSB(ASA21)
; Instance name INSAMP_1, User Module INSAMP
;   Instance name INSAMP_1, Block Name INV(ACA01)
;   Instance name INSAMP_1, Block Name NON_INV(ACA00)
; Instance name INSAMP_2, User Module INSAMP
;   Instance name INSAMP_2, Block Name INV(ACA03)
;   Instance name INSAMP_2, Block Name NON_INV(ACA02)
; Instance name PWM16_1, User Module PWM16
;   Instance name PWM16_1, Block Name PWM16_LSB(DCA04)
        db      33h, 00h        ;PWM16_1_CONTROL_LSB_REG
        db      31h, 37h        ;PWM16_1_PERIOD_LSB_REG
        db      32h, 64h        ;PWM16_1_PWDITH_LSB_REG
;   Instance name PWM16_1, Block Name PWM16_MSB(DCA05)
        db      37h, 00h        ;PWM16_1_CONTROL_MSB_REG
        db      35h, 00h        ;PWM16_1_PERIOD_MSB_REG
        db      36h, 00h        ;PWM16_1_PWDITH_MSG_REG
; Instance name UART_1, User Module UART
;   Instance name UART_1, Block Name RX(DCA07)
        db      3fh, 00h        ;UART_1_RX_CONTROL_REG
        db      3dh, 00h        ;UART_1_
        db      3eh, 00h        ;UART_1_RX_BUFFER_REG
;   Instance name UART_1, Block Name TX(DCA06)
        db      3bh, 00h        ;UART_1_TX_CONTROL_REG
        db      39h, 00h        ;UART_1_TX_BUFFER_REG
        db      3ah, 00h        ;UART_1_
        db      ffh

;End Configuration file trailer End Config.asm
```


Fig. 7C

```
;  PSoCConfig.asm
;
;  This file is generated by the Device Editor on Application Generation.
;  It contains code which loads the configuration data table generated in
;  the file PSoCConfigTBL.asm
;

export LoadConfigInit
export _LoadConfigInit
export LoadConfig_project
export _LoadConfig_project


FLAG_CFG_MASK:          equ     10h             ;M8C flag register REG address bit
mask
END_CONFIG_TABLE:   equ     ffh             ;end of config table indicator

_LoadConfigInit:
 LoadConfigInit:
        lcall   LoadConfig_project

        ret

;
; Load Configuration project
;
_LoadConfig_project:
 LoadConfig_project:
        or      F, FLAG_CFG_MASK                        ;set for
bank 1
        mov     A, >LoadConfigTBL_project_Bank1 ;load bank 1 table
        mov     X, <LoadConfigTBL_project_Bank1
        call    LoadConfig                              ;load the
bank 1 values
        and     F,~FLAG_CFG_MASK                        ;switch
to bank 0
        mov     A, >LoadConfigTBL_project_Bank0 ;load bank 0 table
        mov     X, <LoadConfigTBL_project_Bank0
        call    LoadConfig                              ;load the
bank 0 values
        ret


;
; LoadConfig
;
; This function is not exported.  It assumes that the address of the table
; to be loaded is contained in the X and A registers as if a romx instruction
```


Fig. 8A

```
;;*****************************************************************************
;;*****************************************************************************
;;  ADCINC12.asm
;;
;;  Assembler source for the 12 bit Incremential
;A/D converter.
;;
;;*****************************************************************************
;;*****************************************************************************

export  ADCINC12_1_Start
export _ADCINC12_1_Start
export  ADCINC12_1_SetPower
export _ADCINC12_1_SetPower
export  ADCINC12_1_Stop
export _ADCINC12_1_Stop
export  ADCINC12_1_GetSamples
export _ADCINC12_1_GetSamples
export  ADCINC12_1_StopAD
export _ADCINC12_1_StopAD
export  ADCINC12_1_fIsData
export _ADCINC12_1_fIsData
export  ADCINC12_1_iGetData
export _ADCINC12_1_iGetData
export  ADCINC12_1_ClearFlag
export _ADCINC12_1_ClearFlag

include "ADCINC12_1.inc"
include "m8c.inc"

LowByte:  equ 1
HighByte:  equ 0



;;-----------------------------------------------------------------------
;;  Start:
;;  SetPower:
;;  Applies power setting to the module's analog
;;PSoC block.   Programmable system
;;  INPUTS:  A contians the power setting
;;  OUTPUTS: None.
;;-----------------------------------------------------------------------
 ADCINC12_1_Start:
_ADCINC12_1_Start:
 ADCINC12_1_SetPower:
_ADCINC12_1_SetPower:
  and A,03h
  or  A,f0h
  mov reg[ADCINC12_1_AtoDcr3],A
  ret


;;-----------------------------------------------------------------------
;;  Stop:
;;  SetPower:
;;  Removes power from the module's analog
;;PSoC block.   Programmable system
;;  INPUTS:  None.
;;  OUTPUTS: None.
;;-----------------------------------------------------------------------
 ADCINC12_1_Stop:
_ADCINC12_1_Stop:
  and reg[ADCINC12_1_AtoDcr3], ~03h
  ret


;;-----------------------------------------------------------------------
;;  Get_Samples:
;;  SetPower:
;;  Starts the A/D convertor and will place data is
;;memory.  A flag
;;  is set whenever a new data value is available
;;  INPUTS:  A passes the number of samples (0
;;is continous).
;;  OUTPUTS: None.
;;-----------------------------------------------------------------------
 ADCINC12_1_GetSamples:
_ADCINC12_1_GetSamples:
  mov [ADCINC12_1_bIncrC],A          ;number
;of samples
  or  reg[INT_MSK1],(ADCINC12_1_TimerMask |
ADCINC12_1_CounterMask )
                                     ;Enable both interrupts
  mov [ADCINC12_1_cTimerU],0         ;Force the
;Timer to do one cycle of rest
  or  reg[ADCINC12_1_AtoDcr3],10h  ;force the
;Integrator into reset
  mov [ADCINC12_1_cCounterU],ffh   ;Initialize
;Counter

  mov reg[ADCINC12_1_TimerDR1],ffh
  mov reg[ADCINC12_1_CounterDR1],ffh
  mov reg[ADCINC12_1_TimerCR0],01h ;enable
;the Timer
  mov [ADCINC12_1_fIncr],00h       ;A/D Data
;Ready Flag is reset
  ret
```


Fig. 94

1300

```
;-----------------------------------------
;Interrupt Vector Table
;-----------------------------------------
;
; Interrupt vector table entries are 4 bytes long
;and contain the code
; that services the interrupt (or causes it to be
;serviced).
;
;-----------------------------------------


   AREA   TOP(ROM, ABS)

   org 0                ; Reset Interrupt Vector
   jmp __start           ; First instruction
;executed following a Reset

   org 04h              ; Supply Monitor Interrupt
;Vector
   // call  void_handler
   reti

   org 08h              ; PSoC Block DBA00
;Interrupt Vector
1305 ljmp   ADCINC12_1_TMR_INT
   reti

   org 0Ch              ; PSoC Block DBA01
;Interrupt Vector
1305 ljmp   ADCINC12_1_CNT_INT
   reti

   org 10h              ; PSoC Block DBA02
;Interrupt Vector
   // call  void_handler
   reti

   org 14h              ; PSoC Block DBA03
;Interrupt Vector
   ljmp   Counter16_1INT
   reti

   org 18h              ; PSoC Block DCA04
;Interrupt Vector
   // call  void_handler
   reti

   org 1Ch              ; PSoC Block DCA0
;Interrupt Vector
   ljmp   PWM16_1INT
   reti

   org 20h              ; PSoC Block DCA06
;Interrupt Vector
   ljmp   UART_1TX_INT
   reti

   org 24h              ; PSoC Block DCA07
;Interrupt Vector
1305 ljmp   UART_1RX_INT
   reti

   org 28h              ; Analog Column 0
;Interrupt Vector
   // call  void_handler
   reti

   org 2Ch              ; Analog Column 1
;Interrupt Vector
   // call  void_handler
   reti

   org 30h              ; Analog Column 2
;Interrupt Vector
   // call  void_handler
   reti

   org 34h              ; Analog Column 3
;Interrupt Vector
   // call  void_handler
   reti

   org 38h              ; GPIO Interrupt Vector
   // call  void_handler
   reti

   org 3Ch              ; Sleep Timer Interrupt
;Vector
   jmp SleepTimerISR
   reti
```


Fig. 13A

1350

1351 boot.asm Interrupt Name	1352 Data Sheet Interrupt Name	1353 Type
__start	Reset	Fixed
Interrupt1	Supply Monitor	Fixed
Interrupt2	DBA00	~~PSoC~~ Block → Programmable System
Interrupt3	DBA01	~~PSoC~~ Block → Programmable System
Interrupt4	DBA02	~~PSoC~~ Block → Programmable System
Interrupt5	DBA03	~~PSoC~~ Block → Programmable System
Interrupt6	DCA04	~~PSoC~~ Block → Programmable System
Interrupt7	DCA05	~~PSoC~~ Block → Programmable System
Interrupt8	DCA06	~~PSoC~~ Block → Programmable System
Interrupt9	DCA07	~~PSoC~~ Block → Programmable System
Interrupt10	Analog Column 0	~~PSoC~~ Block → Programmable System
Interrupt11	Analog Column 1	~~PSoC~~ Block → Programmable System
Interrupt12	Analog Column 2	~~PSoC~~ Block → Programmable System
Interrupt13	Analog Column 3	~~PSoC~~ Block → Programmable System
Interrupt14	GPIO	Fixed
Interrupt15	Sleep Timer	Fixed

Fig. 13B